(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 March 2002 (28.03.2002)

PCT

(10) International Publication Number
WO 02/25501 A2

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US01/29787

(22) International Filing Date:
24 September 2001 (24.09.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/667,737 22 September 2000 (22.09.2000) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 09/667,737 (CON)
Filed on 22 September 2000 (22.09.2000)

(71) Applicant *(for all designated States except US)*: **EMATION, INC.** [US/US]; Cabot Business Park, 89 Forbes Boulevard, Mansfield, MA 02048 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **HANSEN, James, R.** [US/US]; 66 Stoneridge Road, Franklin, MA 02038 (US).

(74) Agent: **PYSHER, Paul, A.**; Fish & Richardson, P.C., 225 Franklin Street, Boston, MA 02110 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 02/25501 A2

(54) Title: RETRIEVING DATA FROM A SERVER

(57) Abstract: A system includes a server and a controller embedded in a device. Both the server and the embedded controller are capable of communicating over a computer network. The embedded controller sends a command to the server over the computer network that identifies an instance of the device. In response, the server identifies the instance of the device based on the command, retrieves data that is specific to the instance of the device, and sends the data to the embedded controller over the computer network.

RETRIEVING DATA FROM A SERVER

Background

This invention relates to a controller embedded in a device (an "embedded controller") that retrieves data from a remote server for a specific instance of the device.

A device may contain an embedded controller, such as a microprocessor, to monitor and control its operation. Any type of device may have an embedded controller, including, but not limited to, home appliances, such as washing machines, dishwashers, and televisions, and manufacturing equipment, such as robotics, conveyors and motors.

Embedded controllers, also referred to as "embedded devices", are often connected to an internal network, such as a local area network (LAN), with an interface to the Internet. Other devices on the internal network may communicate with the embedded controllers over the internal network. However, the embedded controllers are not generally addressable from the Internet.

Summary

In general, in one aspect, the invention is directed to a controller embedded in a device for retrieving data from a server. The controller sends a command to the server that identifies an instance of the device and receives, from the server and in response to command, data that is specific to the instance of the device.

This aspect of the invention may include one or more of the following. The command may include an operational parameter for the device and the data may include an updated value for the operational parameter. The command may include plural operational parameters for the device and the data may include updated values that differ from current values of the operational parameters.

The data may include a list of operational parameters. In this case, the embedded controller sends a second command to the server, which includes operational parameters from the list, and receives, from the server and in response to second command, updated values of one or more of the operational parameters included in the second command. The data may include a list of operations to be performed by the

controller. In this case, the embedded controller parses the operations from the list and performs the operations from the list.

The data may include a configuration file for the device. The command may identify the instance of the device by a device type and/or one or more of a serial number and a universal unique identifier. The embedded controller may send the command to the server periodically. The server may run the Hypertext Transfer Protocol and the command may contain Extensible Markup Language code.

In general, in another aspect, the invention is directed to a server for sending data over a network to a controller embedded in a device. The server receives a command from the embedded controller, identifies an instance of the device from information in the command, retrieves data that is specific to the instance of the device, and sends the data to the embedded controller.

This aspect of the invention may include one or more of the following features. The command may include a device type and/or one or more of a serial number and a universal unique identifier. The instance of the device may be identified based on the device type and/or one or more of the serial

number and the universal unique identifier. The server may parse the device type and one or more of the serial number and universal unique identifier from the command prior to identifying the instance of the device.

The command may include an operational parameter for the device. The data may include an updated value of the operational parameter. The data may include a list of operational parameters for the device. The server receives a second command from the embedded controller, which includes an operational parameter from the list of operational parameters, obtains an updated value of the operational parameter, and sends the updated value of the operational parameter to the embedded controller.

The data may include a list of operations to be performed by the embedded controller. The data may include a configuration file for the device. The server may receive the data specific to the instance of the device and store the data in memory, from which it is retrieved. The data specific to the instance of the device may be received via a Web page generated by the server. The server may run the Hypertext Transfer Protocol and the command may contain Extensible Markup Language code.

In general, in another aspect, the invention is directed to a system that includes a controller embedded in a device that is capable of communicating over a computer network, and a server that is capable of communicating over the computer network. The embedded controller sends a command to the server over the computer network that identifies an instance of the device and, in response, the server (i) identifies the instance of the device based on the command, (ii) retrieves data that is specific to the instance of the device, and (iii) sends the data to the embedded controller over the computer network.

This aspect of the invention may include one or more of the following features. The embedded controller is not remotely-addressable from the computer network. The computer network is the Internet. The server runs the Hypertext Transfer Protocol and the command may contain Extensible Markup Language code.

Other features and advantages of the invention will become apparent from the following description, including the claims and drawings.

Brief Description of the Drawings

Fig. 1 is a block diagram of a network containing a server and a device having an embedded controller;

Fig 2 is a flowchart showing a process by which the embedded controller retrieves data for the device from the server; and

Fig 3 is a flowchart showing an alternative process by which the embedded controller retrieves data for the device from the server.

Description

Fig. 1 shows a network 10. Network 10 includes a device 11 containing an embedded controller 17. Device 11 is any type of apparatus or system having functions that are monitored and controlled by embedded controller 17.

Device 11 is connected to an internal network 12, such as a LAN. A router or modem 14 couples internal network 12 to an external network 15, such as the Internet/World Wide Web (Web). External network 15 runs TCP/IP (Transmission Control Protocol/Internet Protocol) or some other suitable protocol. Network connections are via Ethernet, telephone line, wireless, or other transmission media.

External network 15 contains a server 19, which is a computer or any other processing device. Server 19 communicates with embedded controller 17 over external network 15 and internal network 12. Embedded controller 17 has a local IP (Internet Protocol) address that can be resolved within internal network 12. However, this local IP address may not be recognizable by devices on external network 15, such as server 19. As such, server 19 may not be able to directly address device 11.

Embedded Controller

Embedded controller 17 runs software 20, which includes web client application 21 and operating software 22. Web client application 21 includes a TCP/IP protocol stack that allows embedded controller 17 to communicate over external network 15. Device operating software 22 provides an interface between Web client application 21 and a database 24. Through device operating software 22, embedded controller 17 retrieves data stored in database 24 and stores data in database 24.

Database 24 is stored in a memory 25 on device 11 or internal to embedded controller 17. Database 24 stores data,

including operational parameters, configuration files, and identification information for device 11.

The operational parameters constitute settings and/or control instructions for the device 11, which are implemented by embedded controller 17. The types of operational parameters that are stored in database 24 depend on the nature of device 11. For example, if device 11 is a heating/cooling system, the operational parameters may include temperature levels, humidity levels, airflow controls, vent/duct open/close controls, and fan motor speed settings. A configuration file is a file that contains a set of one or more operational parameters for an instance of device 11.

What is meant by "instance" is the specific identity of device 11 as distinguished from other identical devices. The identification information stored in database 24 identifies the instance of device 11. This identification information includes, but is not limited to, data identifying the type of the device, a common (or "friendly") name for the device, the manufacturer of the device, the model name of the device, the model number of the device, the serial number of the device, and a universal unique identifier (UUID) for the device.

The device type specifies a uniform resource locator

(URL) for the device, which includes the name of the device. This information identifies a Web site that is associated with, and generated by, server 19 for the device. For example, a device type might be:

www.SonyVideo.com/television/Vega/XBR400

for a Sony® Vega® XBR400® television that includes an embedded controller. The common name of the device is how the device is known in the vernacular, e.g., "television". The manufacturer identifies the manufacturer of the device, e.g., Sony®. The model name identifies the particular model of the device, e.g., Vega®. The model number identifies the model number of the device, e.g., XBR400®. The serial number identifies the serial number of a particular instance of the device, e.g., 53266D. The UUID is a universal identifier for the instance of the device, e.g., 4A89EA70-73B4-11d4-80DF-0050DAB7BAC5. Of the data shown above, only the serial number and the UUID are unique to the instance of device 11.

Server

Server 19 is a computer that runs HTTP (Hypertext

Transfer Protocol). Server 19 includes a controller 27, such as a microprocessor, for executing software to perform the functions described below. To avoid confusion in terminology, the following reads as though those functions are performed by server 19, even though software in controller 27 of server 19 performs the functions.

Server 19 executes Web server software 29 to communicate over external network 15. Web server software 29 also hosts a Web page associated with device 11. The Web page (not shown) is displayed on the computer of a user, such as the owner of device 11, who may input updated operational parameters for the device. These input updated operational parameters are transmitted to Web server software 29 over external network 15. Web server software 29 stores the updated parameters in database 30.

Web server software 29 stores and retrieves data in database 30 using application logic 32. Application logic 32 is software for accessing database 30 using the CGI (Common Gateway Interface) protocol. CGI is a well-known protocol for accessing a database. The operational parameters can be stored in database 30 individually or as part of a configuration file for an instance of device 11.

Database 30 is stored in a memory 31, which is inside of, or external to, server 19. Database 30 stores data associated with device 11, including the operational parameters noted above. Other data that may be stored for device 11 is described below.

The Data Transfer Process

Embedded controller 17 executes software 20 to retrieve data, such as operational parameters, from remote server 19. Server 19 executes software 34 to send the data to embedded controller 17. Fig. 2 shows these processes in detail. The left half of Fig. 2, titled "Embedded Controller" shows process 40 performed by embedded controller 17, and the right half of Fig. 2, titled, "Server", shows process 41 performed by server 19.

Process 40 generates and sends (201) a command to server 19. The command, or a modified version thereof, is sent by embedded controller 17 to server 19 periodically. It is through this command that embedded controller 17 polls server 19 to determine if there are any new/updated operational parameters for device 11.

The command includes data identifying device 11. The

data identifies the specific instance of device 11 and includes a device type field and one or both of a device serial number field and a device UUID. The command may also include the common name field, the manufacturer name field, the model name field, and the model number field, as set forth above.

The command may be either an HTTP GET command or an HTTP post command. The data included in those commands is similar, with the difference being that the HTTP GET command retrieves a document, such as a configuration file, that contains operational parameters and the HTTP POST command retrieves individual operational parameters. An example of an HTTP GET command is shown in Appendix A and an example of an HTTP POST command is shown in Appendix B.

The HTTP POST and GET commands shown in Appendices A and B contain XML (eXtensible Markup Language) commands. XML is a self-describing computer language in the sense that fields in the XML code identify variables and their values in the XML code. For example, as shown in the Appendices, the "manufacturer" field identifies a manufacturer, e.g., Sony®, and is delineated by "<manufacturer>" to indicate the start of the field and "</manufacturer>" to indicate the end of the

field. XML is used because it can be generated, parsed and read relatively easily by server 19 and embedded controller 17.

As noted, the GET command is used to retrieve a document from server 19. The document to be retrieved corresponds to the fields in the GET command, in particular to the device type, serial number and/or UUID fields. By contrast, the POST command is used to retrieve individual operational parameters. The operational parameters that are to be retrieved are listed in the POST command itself. For example, as shown in Appendix B, the operational parameters include airflow, humidity, motor and vent values for the fictitious "widget" device. The current values of these parameters are specified in the POST command shown in Appendix B as follows:

```
<parameters>
  <Airflow xsd:type="integer">378</Airflow>
  <Humidity xsd:type="double">46.7</Humidity>
  <Motor xsd:type="integer">1500</Motor>
  <Vent xsd:type="integer">4</Vent>
</parameters>
```

The updated values of these parameters are returned by server 19 to embedded controller 17 in a reply POST command. The updated values of these parameters are specified in the POST

command shown in Appendix B as follows:

```
<parameters>
  <Motor xsd:type="integer">1250</ Motor >
  <Vent xsd:type="integer">2</Vent>
</parameters>
```

As shown, both the POST and GET commands include the URL of the device in the device type field. As described below, this directs server 19 to a Web site associated with device 11 and, thereafter, in the case of a GET Command, to retrieve a specific Web page that is generated by server 19 for the device. It is noted that, since the POST command retrieves parameters, not a document like the GET command, the POST command need not include a URL of the device.

Referring back to Fig. 2, process 41 (in server 19) receives (202) the command from embedded controller 17. Process 41 identifies the command as either a POST or GET command based on a header, such as "POST/CONTROL HTTP/1.1" (see the headers in Appendices A and B), in the command. Process 41 uses an XML parser to parse (203) the various identifying fields, such as device type, serial number, and UUID, from the command.

Process 41 identifies (204) the instance of device 11

based on the information parsed from the command. That is, process 41 uses the device type, serial number, and UUID field information to identify the instance.

If The Command Is A POST Command

The remaining identification information from the command is used to narrow the search through database 30 down to data for the specific instance of device 11. The device serial number and/or UUID are used to retrieve operational parameters specific to device 11.

Once the appropriate data has been identified (204), process 41 retrieves (205) that data using application logic 32. Process 41 compares the values of the operational parameters to those included in the POST command. If the values are the same, process 41 returns an indication that there are no new/updated values for device 11. If the values of the operational parameters are different, process 41 adds the appropriate updated value fields to the POST command and sends (206) the POST command, with the updated operational parameters, back to embedded controller 17. Thus, only those operational parameters that differ from their original values are returned to embedded controller 17 in the POST command.

If The Command Is A GET Command

As was the case above with the POST command, the remaining identification information from the command is used to narrow the search through database 30 down to data for the specific instance of device 11. In particular, the device serial number and/or UUID are used to retrieve (205) a configuration file that is specific to device 11. Process 41 then sends (206) the configuration file to embedded controller 17. The configuration file may be a Web page identified by the URL in the device type field. This Web page is generated by server 19 using parameters stored in database 30 and then sent to device 11. It is noted that the complete Web page itself need not be stored. Alternatively, the GET command may retrieve separate configuration files and Web pages.

Process 40 in embedded controller 17 receives (207) the data (operational parameters or configuration file) from server 19 in response to sending (201) the command. Process 40 then uses the data to update/reset device 11. For example, if device 11 is a heating system, a new operational parameter may be a new temperature setting for its thermostat. In this

example, embedded controller 17 sets the new temperature accordingly. If the device is a television, a new operational parameter may indicate that certain pay television stations are now available. In this case, embedded controller 17 performs any appropriate decoding/descrambling functions on the television signal.

Alternative Embodiment

Fig. 3 shows alternative embodiments of processes 40,41. In processes 40,41, the GET and POST commands request the same parameters each time the commands are issued. The parameters requested are encoded in the software to implement process 40. This embodiment provides a way to change the parameters that are requested without altering the software that generates the request/command.

Referring to Fig. 3, process 45 in embedded controller 17 begins by sending (301) a command to server 19. The command, in this case, is an HTTP GET command, since it is requesting a document, not individual operational parameters. The document is an XML document that contains a list of operational parameters to be updated. Using this document, embedded controller 17 can change the operational parameters that it

periodically updates.

Process 46 in server 19 receives (302) the command from embedded controller 17, parses (303) the command using an XML parser to obtain the information specific to the instance of device 11, and identifies (304) the appropriate document based on this information. As before, the information that identifies the instance of device 11 includes, among other things, the device type, its serial number, and its UUID. Process 46 retrieves (305) the document containing the list of operational parameters to be updated, and sends (306) the document back to embedded controller 17.

Process 45 in embedded controller 17 receives (307) the document from server 19, parses (308) the operational parameters to be updated from the document, and formulates (309) a POST command to send to server 19. The command is formulated using a command template (not shown), into which process 45 inserts the operational parameters parsed from the document. Process 45 sends this second command to the server. At this point, processes 45 and 46 operate (310) in the same manner as processes 40 and 41, respectively, when used with a POST command. Accordingly, the details of processes 40,41 are not repeated here.

This alternative embodiment may be generalized further. For example, rather than simply retrieving a list of operational parameters, embedded controller 17 may retrieve, from server 19, a list of operations that it is to perform. For example, that list may contain operational parameters to be updated, times at which the updates are to occur, a schedule of diagnostic tests, and the like. Any operation that may be performed by embedded controller 17 may be included on the list.

The process for retrieving the list of operations is identical to processes 45 and 46, save for the contents of the list itself. The actions that embedded controller takes once it has the list (i.e., 310) depend on the contents of the list. For example, the list might specify that parameters are to be updated every hour and may also contain a list of the parameters to be updated. The list may contain XML commands, which can be parsed by embedded controller 17. Thus, embedded controller 17 reads the commands in the list and performs the appropriate operations with respect to device 11.

Architecture

Processes 40,41 and 45,46 are not limited to use with the

hardware/software configuration of Fig. 1; they may find applicability in any computing or processing environment. Processes 40,41 and 45,46 may be implemented in hardware (e.g., an ASIC {Application-Specific Integrated Circuit} and/or an FPGA {Field Programmable Gate Array}), software, or a combination of hardware and software.

Processes 40,41 and 45,46 may be implemented using one or more computer programs executing on programmable computers that each includes a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and one or more output devices.

Each such program may be implemented in a high level procedural or object-oriented programming language to communicate with a computer system. Also, the programs can be implemented in assembly or machine language. The language may be a compiled or an interpreted language.

Each computer program may be stored on a storage medium or device (e.g., CD-ROM, hard disk, or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform

processes 40,41 and 45,46.

Processes 40,41 and 45,46 may also be implemented as a computer-readable storage medium, configured with a computer program, where, upon execution, instructions in the computer program cause the computer to operate in accordance with processes 40,41 and 45,46.

The invention is not limited to use with the protocols and standards described above. For example, Web server may use Java Servlets, ASP (Active Server Pages), and/or ISAPI (Internet Server Application Programming Interface) to communicate with application logic 32, instead of, or in addition to, CGI. The commands sent by embedded controller 17 and/or server 19 (e.g., in 201, 301, 310) are not limited to HTTP GET and POST commands. Any commands and/or requests for requesting and receiving data may be used.

The data transferred to embedded controller 17 by server 19 is not limited to operational parameters or configuration files. The data may include, for example, a schedule of actions to be performed by device 11 that is based on information pertaining the owner of the device. For example, owner preferences may be stored in database 30. The instance-specific data may be used by server 19 to correlate the owner

of the device to the appropriate preferences. These preferences then may be transmitted back to device 11 to control the operation thereof.

The original parameters sent by embedded controller 17 to server 19 may be used by server 19 to calculate new, updated parameters based on data stored in database 30. Thus, the invention is not limited to simply retrieving updated data, but may also include calculating new data based on currently-available data.

The documents and commands described above are not limited to XML format. Any computer language may be used for the commands. The documents may be in any format, for example, HTML (Hypertext Markup Language) documents may be used. In addition, the invention is not limited to use with the Web, Web servers, and the like. The servers and embedded controllers described herein may be the same type of general-purpose computer appropriately programmed, or different devices.

Other embodiments not described herein are also within the scope of the following claims.

Appendix A

GET COMMAND

```
GET /Widget/config.xml HTTP/1.1
HOST: www.acme.com
Content-Type: text/xml
Content-length: nnn

<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
    <deviceType>urn:www-acme-com:device:Widget:3</deviceType>
    <friendlyName>Widget</friendlyName>
    <manufacturer>Acme Industries</manufacturer>
    <modelName>Widget</modelName>
    <modelNumber>3</modelNumber>
    <serialNumber>53266D</serialNumber>
    <UDN>uuid:4A89EA70-73B4-11d4-80DF-0050DAB7BAC5</UDN>
  </device>
</root>
```

Appendix B

POST COMMAND

```
POST /CONTROL HTTP/1.1
Host: www.acme.com
Content-Type: text/xml
Content-length: nnn

<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
    <deviceType>urn:www-acme-com:device:Widget:3</deviceType>
    <friendlyName>Widget</friendlyName>
    <manufacturer>Acme Industries</manufacturer>
    <modelName>Widget</modelName>
    <modelNumber>3</modelNumber>
    <serialNumber>53266D</serialNumber>
    <UDN>uuid:4A89EA70-73B4-11d4-80DF-0050DAB7BAC5</UDN>
  </device>
</root>
<parameters>
  <Airflow xsd:type="integer">378</Airflow>
  <Humidity xsd:type="double">46.7</Humidity>
  <Motor xsd:type="integer">1500</Motor>
  <Vent xsd:type="integer">4</Vent>
</parameters>
```

And the response containing parameters that have been modified:

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Content-length: nnn
Date: Fri, 13 Jun 2000 13:43:05 GMT

<?xml version="1.0"?>
```

```
<parameters>
  <Motor xsd:type="integer">1250</ Motor >
  <Vent xsd:type="integer">2</Vent>
</parameters>
```

What is claimed is:

1. A method performed by a controller embedded in a device for retrieving data from a server, comprising:

sending a command to the server that identifies an instance of the device; and

receiving, from the server and in response to command, data that is specific to the instance of the device.

2. The method of claim 1, wherein the command includes an operational parameter for the device and the data comprises an updated value for the operational parameter.

3. The method of claim 2, wherein the command includes plural operational parameters for the device and the data comprises updated values that differ from current values of the operational parameters.

4. The method of claim 1, wherein the data comprises a list of operational parameters; and

the method further comprises:

sending a second command to the server, which includes operational parameters from the list; and

receiving, from the server and in response to second command, updated values of one or more of the operational parameters included in the second command.

5. The method of claim 1, wherein the data comprises a list of operations to be performed by the controller; and

the method further comprises:

parsing the operations from the list; and

performing the operations from the list.

6. The method of claim 1, wherein the data comprises a configuration file for the device.

7. The method of claim 1, wherein the command identifies the instance of the device by a device type and/or one or more of a serial number and a universal unique identifier.

8. The method of claim 1, wherein the embedded controller sends the command to the server periodically.

9. The method of claim 1, wherein the server runs the Hypertext Transfer Protocol and the command contains Extensible Markup Language Code.

10. A method performed by a server for sending data over a network to a controller embedded in a device, comprising:

receiving a command from the embedded controller;

identifying an instance of the device from information in the command;

retrieving data that is specific to the instance of the device; and

sending the data to the embedded controller.

11. The method of claim 10, wherein:

the command includes a device type and/or one or more of a serial number and a universal unique identifier; and

the instance of the device is identified based on the device type and/or one or more of the serial number and the universal unique identifier.

12. The method of claim 11, further comprising:

parsing the device type and one or more of the serial number and universal unique identifier from the command prior to identifying the instance of the device.

13. The method of claim 10, wherein:

the command includes an operational parameter for the device; and

the data comprises an updated value of the operational parameter.

14. The method of claim 10, wherein:

the data comprises a list of operational parameters for the device; and

the method further comprises:

receiving a second command from the embedded controller, which includes an operational parameter from the list of operational parameters;

obtaining an updated value of the operational parameter; and

sending the updated value of the operational parameter to the embedded controller.

15. The method of claim 10, wherein the data comprises a list of operations to be performed by the device.

16. The method of claim 10, wherein the data comprises a configuration file for the device.

17. The method of claim 10, further comprising:

receiving the data specific to the instance of the device; and

storing the data in memory;

wherein the data is retrieved from the memory.

18. The method of claim 17, wherein the data specific to the instance of the device is received via a Web page generated by the server.

19. The method of claim 10, wherein the server runs the Hypertext Transfer Protocol and the command contains Extensible Markup Language Code.

20. A system comprising:

a controller embedded in a device, the controller being capable of communicating over a computer network; and

a server that is capable of communicating over the computer network;

wherein the embedded controller sends a command to the server over the computer network that identifies an instance of the device and, in response, the server (i) identifies the instance of the device based on the command, (ii) retrieves data that is specific to the instance of the device, and (iii) sends the data to the embedded controller over the computer network.

21. The system of claim 20, wherein the embedded controller is not remotely-addressable from the computer network.

22. The system of claim 20, wherein the computer network comprises the Internet.

23. The system of claim 20, wherein the server runs the Hypertext Transfer Protocol and the command contains Extensible Markup Language Code.

24. A computer program stored on a computer-readable medium, the computer program being executable by a controller embedded in a device to retrieve data from a server, the computer program comprising instructions that cause the embedded controller to:

send a command to the server that identifies an instance of the device; and

receive, from the server and in response to command, data that is specific to the instance of the device.

25. The computer program of claim 24, wherein the command includes an operational parameter for the device and the data comprises an updated value for the operational parameter.

26. The computer program of claim 25, wherein the command includes plural operational parameters for the device and the data comprises updated values that differ from current values of the operational parameters.

27. The computer program of claim 24, wherein the data

comprises a list of operational parameters; and

the computer program further comprises instructions that cause the embedded controller to:

send a second command to the server, which includes operational parameters from the list; and

receive, from the server and in response to second command, updated values of one or more of the operational parameters included in the second command.

28. The computer program of claim 24, wherein the data comprises a list of operations to be performed by the controller; and

the computer program further comprises instructions that cause the embedded controller to:

parse the operations from the list; and

perform the operations from the list.

29. The computer program of claim 24, wherein the data comprises a configuration file for the device.

30. The computer program of claim 24, wherein the command identifies the instance of the device by a device type and/or one or more of a serial number and a universal unique identifier.

31. The computer program of claim 24, wherein the embedded controller sends the command to the server periodically.

32. The computer program of claim 24, wherein the server runs the Hypertext Transfer Protocol and the command contains Extensible Markup Language Code.

33. A computer program stored on a computer-readable medium that is executable by a server to send data over a network to a controller embedded in a device, the computer program comprising instructions that cause the server to:

receive a command from the embedded controller;

identify an instance of the device from information in the command;

retrieve data that is specific to the instance of the device; and

send the data to the embedded controller.

34. The computer program of claim 33, wherein:

the command includes a device type and/or one or more of a serial number and a universal unique identifier; and

the instance of the device is identified based on the device type and/or one or more of the serial number and the universal unique identifier.

35. The computer program of claim 34, further comprising instructions that cause the server to:

parse the device type and one or more of the serial number and universal unique identifier from the command prior to identifying the instance of the device.

36. The computer program of claim 33, wherein:

the command includes an operational parameter for the device; and

the data comprises an updated value of the operational parameter.

37. The computer program of claim 33, wherein:

the data comprises a list of operational parameters for the device; and

the computer program further comprises instructions that cause the server to:

receive a second command from the embedded controller, which includes an operational parameter from the list of operational parameters;

obtain an updated value of the operational parameter; and

send the updated value of the operational parameter to the embedded controller.

38. The computer program of claim 33, wherein the data comprises a list of operations to be performed by the device.

39. The computer program of claim 33, wherein the data comprises a configuration file for the device.

40. The computer program of claim 33, further comprising instructions that cause the server to:

receive the data specific to the instance of the device;

and

store the data in memory;

wherein the data is retrieved from the memory.

41. The computer program of claim 40, wherein the data specific to the instance of the device is received via a Web page generated by the server.

42. The computer program of claim 33, wherein the server runs the Hypertext Transfer Protocol and the command contains Extensible Markup Language Code.

43. An apparatus for retrieving data from a server, comprising:

a memory which stores executable instructions; and

a controller which executes the instructions to:

send a command to the server that identifies an instance of the device; and

receive, from the server and in response to command, data that is specific to the instance of the device.

44. The apparatus of claim 43, wherein the command

includes an operational parameter for the device and the data comprises an updated value for the operational parameter.

45. The apparatus of claim 44, wherein the command includes plural operational parameters for the device and the data comprises updated values that differ from current values of the operational parameters.

46. The apparatus of claim 43, wherein the data comprises a list of operational parameters; and

the apparatus executes instructions to:

send a second command to the server, which includes operational parameters from the list; and

receive, from the server and in response to second command, updated values of one or more of the operational parameters included in the second command.

47. The apparatus of claim 43, wherein the data comprises a list of operations to be performed by the controller; and

the apparatus executes instructions to:

parse the operations from the list; and

perform the operations from the list.

48. The apparatus of claim 43, wherein the data comprises a configuration file for the device.

49. The apparatus of claim 43, wherein the command identifies the instance of the device by a device type and/or one or more of a serial number and a universal unique identifier.

50. The apparatus of claim 43, wherein the embedded controller sends the command to the server periodically.

51. The apparatus of claim 43, wherein the server runs the Hypertext Transfer Protocol and the command contains Extensible Markup Language Code.

52. An apparatus for sending data over a network to a controller embedded in a device, comprising:

a memory which stores executable instructions; and

a controller which executes the instructions to:

receive a command from the embedded controller;

identify an instance of the device from information in the command;

retrieve data that is specific to the instance of the device; and

send the data to the embedded controller.

53. The apparatus of claim 52, wherein:

the command includes a device type and/or one or more of a serial number and a universal unique identifier; and

the instance of the device is identified based on the device type and/or one or more of the serial number and the universal unique identifier.

54. The apparatus of claim 53, wherein the apparatus executes instructions to:

parse the device type and one or more of the serial number and universal unique identifier from the command prior to identifying the instance of the device.

55. The apparatus of claim 52, wherein:

the command includes an operational parameter for the device; and

the data comprises an updated value of the operational parameter.

56. The apparatus of claim 52, wherein:

the data comprises a list of operational parameters for the device; and

the apparatus executes instructions to:

receive a second command from the embedded controller, which includes an operational parameter from the list of operational parameters;

obtain an updated value of the operational parameter; and

send the updated value of the operational parameter to the embedded controller.

57. The apparatus of claim 52, wherein the data comprises a list of operations to be performed by the device.

58. The apparatus of claim 52 wherein the data comprises a configuration file for the device.

59. The apparatus of claim 52, wherein:

the apparatus executes instructions to:

receive the data specific to the instance of the device; and

store the data in memory; and

the data is retrieved from the memory.

60. The apparatus of claim 59, wherein the data specific to the instance of the device is received via a Web page generated by the server.

61. The apparatus of claim 52, wherein the apparatus runs the Hypertext Transfer Protocol and the command contains Extensible Markup Language Code.

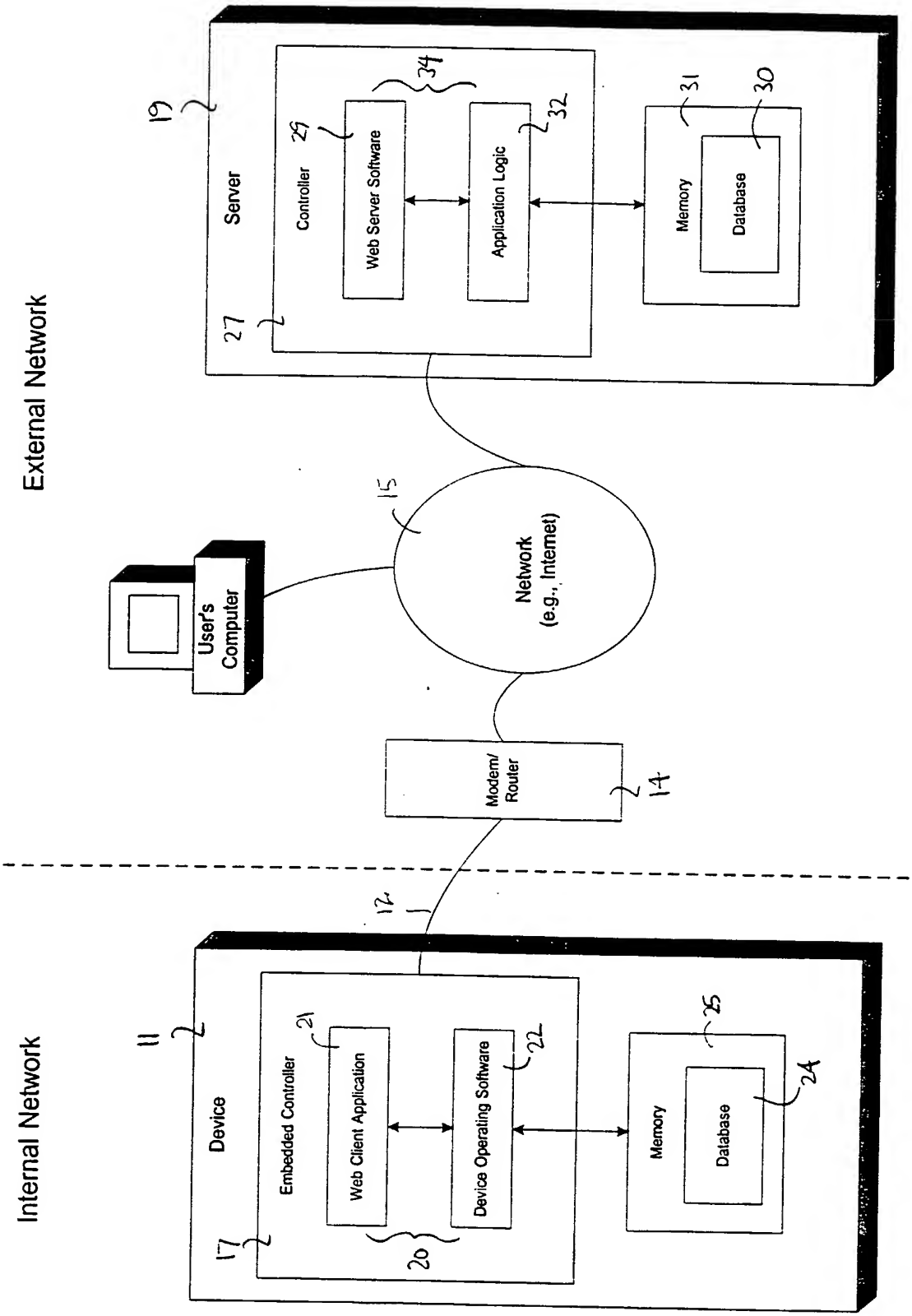
External Network
Internal Network
Server
Controller
Web Server Software
Application Logic
Memory
Database
User's Computer
Network (e.g., Internet)
Modem/ Router
Device
Embedded Controller
Web Client Application
Device Operating Software
Memory
Database
19
27
29
34
32
31
30
15
14
12
11
17
21
22
20
25
24

FIG. 1

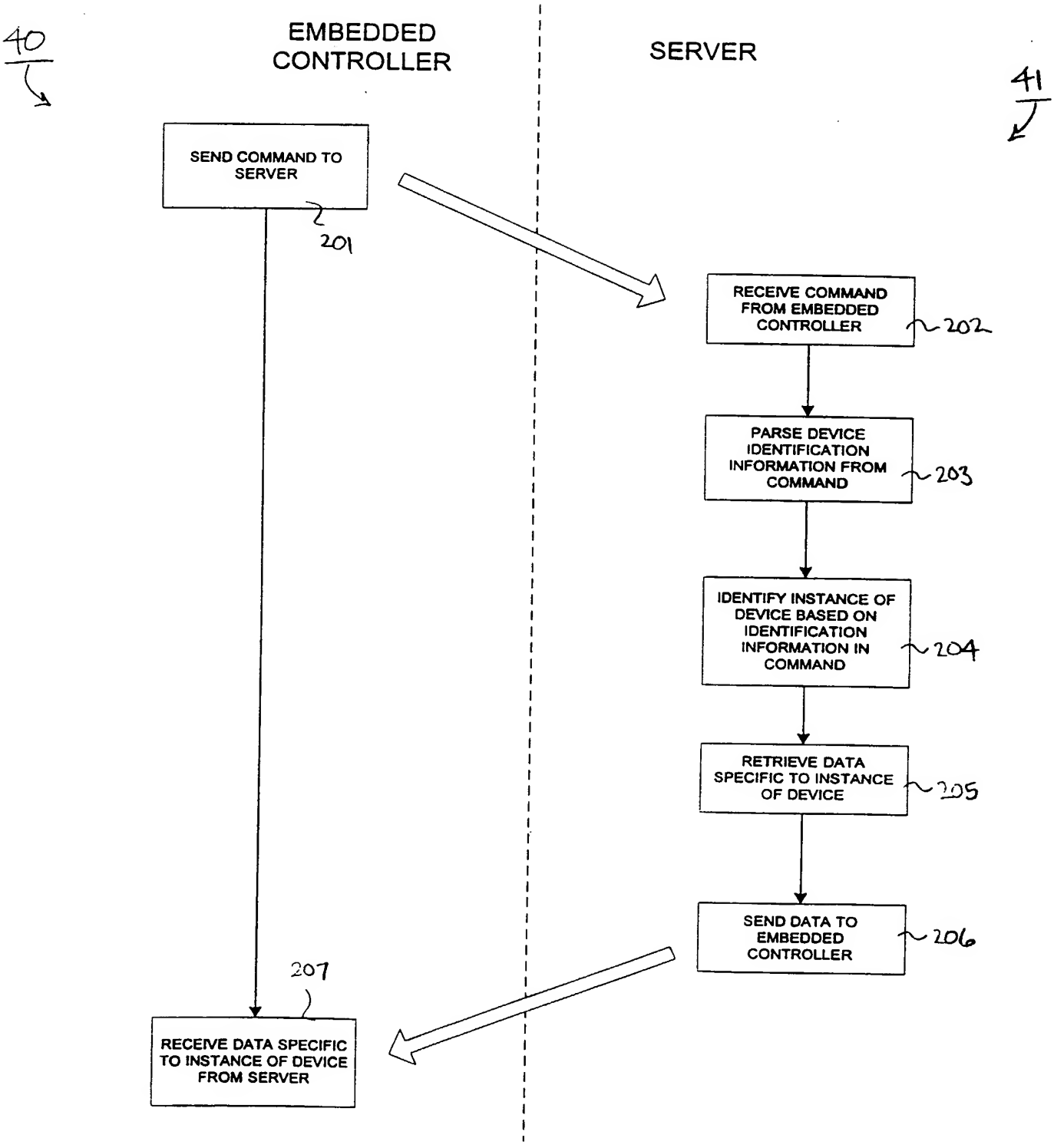
40
EMBEDDED CONTROLLER
SERVER
41
SEND COMMAND TO SERVER
201
RECEIVE COMMAND FROM EMBEDDED CONTROLLER
202
PARSE DEVICE IDENTIFICATION INFORMATION FROM COMMAND
203
IDENTIFY INSTANCE OF DEVICE BASED ON IDENTIFICATION INFORMATION IN COMMAND
204
RETRIEVE DATA SPECIFIC TO INSTANCE OF DEVICE
205
SEND DATA TO EMBEDDED CONTROLLER
206
207
RECEIVE DATA SPECIFIC TO INSTANCE OF DEVICE FROM SERVER

FIG. 2

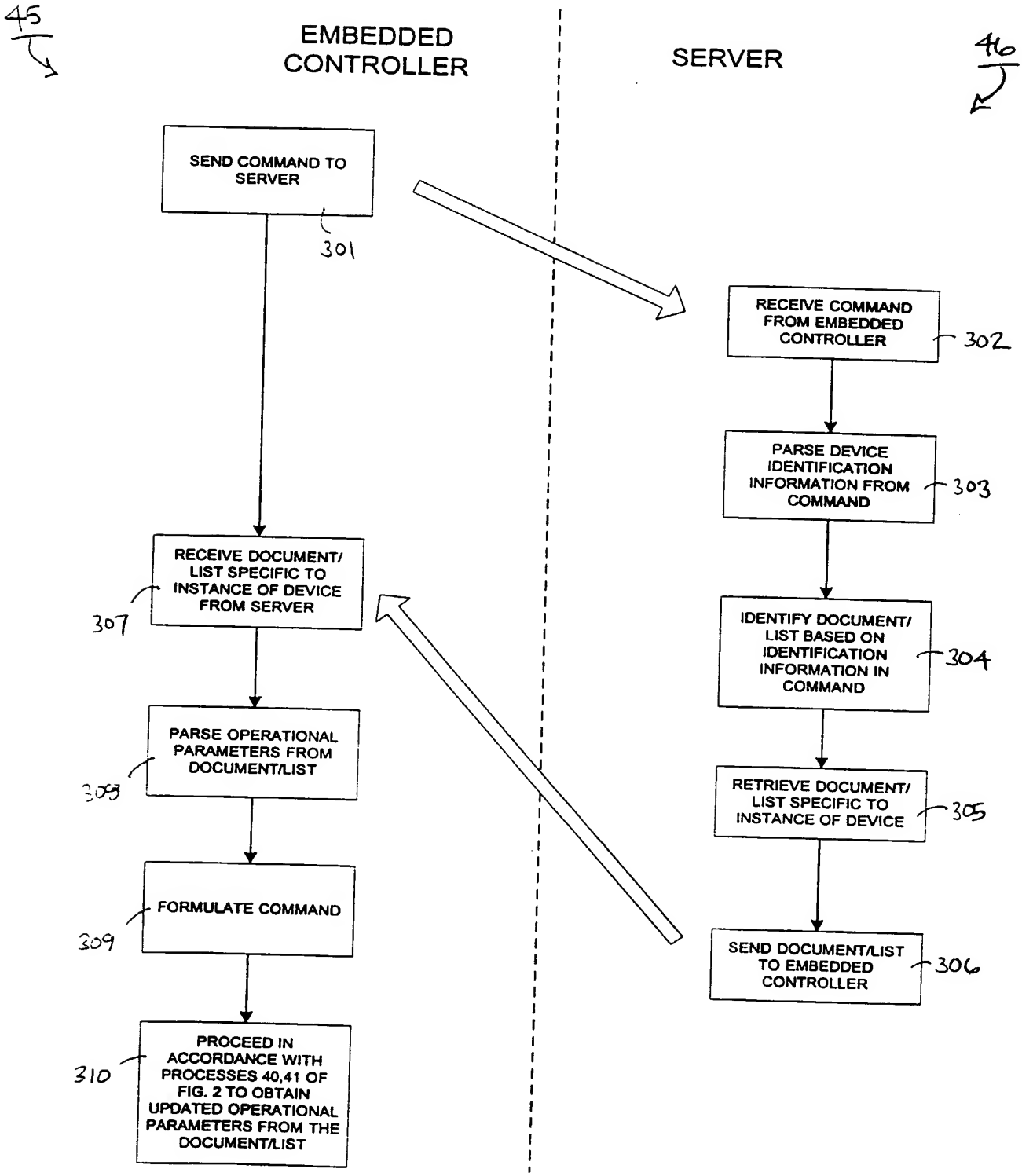
45
EMBEDDED CONTROLLER
SERVER
46
SEND COMMAND TO SERVER
301
RECEIVE COMMAND FROM EMBEDDED CONTROLLER
302
PARSE DEVICE IDENTIFICATION INFORMATION FROM COMMAND
303
IDENTIFY DOCUMENT/ LIST BASED ON IDENTIFICATION INFORMATION IN COMMAND
304
RETRIEVE DOCUMENT/ LIST SPECIFIC TO INSTANCE OF DEVICE
305
SEND DOCUMENT/LIST TO EMBEDDED CONTROLLER
306
RECEIVE DOCUMENT/ LIST SPECIFIC TO INSTANCE OF DEVICE FROM SERVER
307
PARSE OPERATIONAL PARAMETERS FROM DOCUMENT/LIST
308
FORMULATE COMMAND
309
PROCEED IN ACCORDANCE WITH PROCESSES 40,41 OF FIG. 2 TO OBTAIN UPDATED OPERATIONAL PARAMETERS FROM THE DOCUMENT/LIST
310

FIG. 3